JAVA PROGRAMMING LANGUAGE

WEEK-3

Question: Basic element of java?

Ans:

Java programs are structured in a way that promotes code organization and reusability. The basic building block is a class that contains fields (variables) and methods (functions). Here's a detailed breakdown:

Class: A class is a blueprint for objects. It defines the properties (fields) and behaviors (methods) that objects of that class will have. For example, if you were creating a class to represent a Person, it might have fields like name and age, and methods like sayHello().

Fields: Fields are variables that belong to a class. They hold data that represents the state of an object. For instance, in the Person class, name and age would be fields.

Methods: Methods are blocks of code that perform specific tasks. They can be called to execute their functionality. For example, the sayHello() method in the Person class prints a greeting.

```
public class Person {

String name; // Field

 int age; // Field

 void sayHello() { // Method

        System.out.println("Hello, my name is " + name);

         }

}
```

Question Data types?

Ans:

Java supports several data types that can be broadly categorized into primitive and reference types. Understanding these types is crucial for working with data effectively:

Primitive Types: These are the most basic data types in Java. They include integers (int), floating-point numbers (double), characters (char), booleans (boolean), etc.

Reference Types: These include objects, arrays, and interfaces. Objects are instances of classes, arrays are collections of elements of the same type, and interfaces define contracts for classes.

```
int mylnt = 42; // Primitive Type

double myDouble = 3.14; // Primitive Type

char myChar = 'A'; // Primitive Type

boolean myBoolean = true; // Primitive Type

String myString = "Hello, World!"; // Reference Type
```

Question: Operators?

Ans:

Operation are symbols that perform operation on variables or values. Understanding how to use these allows you to manipulate data efficiently:

Arithmetic Operators: These include + (addition), (subtraction), * (multiplication), / (division), % (modulo).

Assignment Operators: These assign values to variables. Examples include =, +=, -=.

Comparison Operators: These are used for comparing values. They include == (equals), != (not equals), >, <, >=, <=.

Logical Operators: These are used for combining boolean expressions. They include && (and), || (or), ! (not).

Bitwise Operators: These operate on individual bits of numbers. They include & (bitwise and), | (bitwise or), ^ (bitwise

```
int a = 5;

int b = 3;

int sum = a + b; // sum is now 8
```

Question: Control statement ?

Ans:

Control statements allows us to control the flow of execution in your program. These are essential for making decisions and repeating tasks:

Conditional Statements: These include if-else statements for making decisions based on conditions, and switch statements for selecting among multiple alternatives.

```
int grade = 85;

if (grade >= 60) {

System.out.println("Passed");

} else {
```

Question: Loops?

Ans:

Loops: Loops allows us to execute a block of code repeatedly. Common types include for, while, and do-while loops.

```
for (int i = 0; i < 5; i++) {

System.out.println("Iteration" + i);

}
```

Question: Methods?

Ans:

Methods are blocks of code that perform specific tasks. They can accept parameters (input) and return values (output). They play a crucial role in organizing and reusing code.

```
public int add(int a, int b) {

return a + b;
```

Question: object?

Ans:

Object: An object is an instance of a class. It encapsulates data (fields) and behavior (methods).

```
public class Person {

String name;

int age;

void sayHello() {

        System.out.println("Hello, my name is " + name);

        }

}

Person john = new Person();

john.name = "John";

john.age = 30;

john.sayHello(); // Output: "Hello, my name is John"
```

Question: Inheritance?

Ans:

Inheritance is a fundamental concept in Object- Oriented Programming (OOP) that allows one class (called the subclass) to inherit the attributes and methods

of another class (called the superclass). It promotes code reusability and allows for the creation of hierarchies of classes.

Superclass: The class that is being extended or inherited from is called the superclass.

Subclass: The class that is inheriting from the superclass is called the subclass.

```
public class Person {

String name;

int age;

void sayHello() {

System.out.println("Hello, my name is " + name);
```

Question: Interface and abstract ?

Ans:

Interfaces and abstract classes are used to define contracts for classes. It allows us to specify the methods that implementing classes must provide.

Interface: An interface defines a contract of methods that a class must implement. It does not provide any implementation details, only method signatures.

```
Public interface Shape {

double calculateArea();

}
```

In this example, the Shape interface declares a method calculateArea(). Any class that implements this interface must provide an implementation for calculateArea().

Abstract Class: An abstract class is similar to an interface, but it can have both implemented and abstract methods. An abstract method is a method without an implementation.

```
public abstract class Shape2D implements Shape {

int sides;
```

```
abstract double calculate Perimeter();

}
```

In this example, Shape2D is an abstract class that extends the Shape interface. It introduces an abstract method calculate Perimeter() that subclasses must implement.

Question : Package and modules?

Ans:

Packages and modules are used to organize code into manageable units. They are essential for organizing large projects and preventing naming conflicts.

Package: A package is a way to organize related classes, interfaces, enumerations, and annotations. It helps to avoid naming conflicts and provides a clear hierarchy.

```
Package com.example.myproject;

public class MyClass {

// ...

}
```

In this example, com.example.myproject is the package name, and MyClass belongs to this package.

Question: Exception handling ?

Ans:

Packages and modules are used to organize code into manageable units. They are essential for organizing large projects and preventing naming conflicts.

Package: A package is a way to organize related classes, interfaces, enumerations, and annotations. It helps to avoid naming conflicts and provides a clear hierarchy.

```
Package com.example.myproject;

public class MyClass {

// ...
```

```
}
```

In this example, com.example.myproject is the package name, and MyClass belongs to this package.

Question: Polymorphism?

Ans:

Polymorphism is a core principle of Object- Oriented Programming that allows objects to take on multiple forms. There are two main types of polymorphism in Java:

Compile-time Polymorphism (Static Binding): It is achieved through method overloading and method overriding.

Method Overloading: It allows a class to have multiple methods with the same name, but with different parameter lists. The correct method to call is determined at compile-time based on the arguments provided.

```
Public class Animal {

Public void makeSound() {

        System.out.println("Some generic sound");

        }

}

Public class Dog extends Animal {

 @Override

public void makeSound() {

System.out.println("Bark!");

        }

}
```